IN THE SPECIFICATION:

Please amend the specification as follows:

Page 1, lines 21 - 29:

With this in mind, semiconductor tester architectures usually fall within one of two types, shared resource, or tester-per-pin. In a conventional tester-per-pin architecture, such as that shown in Figure 1, each tester channel 10 includes separate tester resources for assignment to one pin of a DUT. Those resources include the necessary timing circuitry 12, pattern generation or data circuitry 14, formatting circuitry 16, and pin electronics 18 for applying signals to, or receiving signals from a pin of the DUT 20. A failure processing circuit handles the evaluation of expected test data versus actual data. This architecture is highly desirable when testing complex logic devices that need flexibility in testing the individual device pins.

Page 4, lines 2 - 13:

The invention will be better understood by reference to the following more detailed description and accompanying drawings in which

FIG. 1 is a high-level block diagram of a conventional per-pin tester architecture;

FIG. 2 is a high-level block diagram of a conventional shared resource tester architecture; ~~and~~

FIG. 3 is a high-level block diagram of a hybrid tester architecture according to one form of the present invention; and

FIG. 4 is a flowchart illustrating a method of testing a plurality of semiconductor devices with the hybrid tester architecture of Figure 3.

Page 5, lines 30 - 33:

In operation, and ~~Consistent~~ consistent with the example above, each of the per-pin memory blocks generates test data signals unique to each pin, at step 200 (Fig. 4), and feeds its output to a corresponding per-pin formatting circuit 110 (for a total of 2048 blocks). As noted above, the per-pin data gets clocked through the per-pin formatting circuitry by the shared-pin timing circuitry 108, at step 202.

Page 6, lines 4 - 8:

As testing progresses, the test signals are applied to each of the DUTs 106, at step 204, ~~undergoes write and read operations~~ so that the tester can determine if any of the DUTs have faulty cells, and if so, where those cells lie. The failure data for each DUT 106 is routed through respective databuses 111 (shown in phantom) and stored in the failure memory 115, at step 206, as is well known in the art.

Page 6, lines 9 - 18:

Following completion of the test, the failure data from the failure memory 115 is fed to the on-tester redundancy analysis circuitry 117. The circuitry analyzes the faulty data, at step 208, for each DUT and assigns redundant rows and columns (built into each DUT) to "fix" the device. Each redundancy solution is unique for a particular DUT (the redundant row/column addresses) and generally involves programming the DUT to incorporate certain row and column address lines. For on-tester device repair, the solution data is fed back to the computer workstation 102. Once the solutions are known, corresponding test patterns are created and fed to the per-pin data resources 112 to then generate write addresses for programming activation of the desired redundancy elements for each device.